AD NUMBER _____

TECOM PROJECT NO. 7-CO-R88-EPO-008

PUBLICATION NO. USAEPG-FR-1373, Vol II

FINAL REPORT

METHODOLOGY INVESTIGATION

OF

AI TEST OFFICER SUPPORT TOOL

VOLUME II

BY

ROBERT HARDER

Software and Interoperability Division
Electronic Technology Test Directorate

US ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, ARIZONA 85613-7110

MARCH 1989

PREPARED FOR: CDR, TECOM

PERIOD COVERED: OCTOBER 1987 - OCTOBER 1988

Approved for public release; distribution unlimited.

US Army Test and Evaluation Command
Aberdeen Proving Ground, MD 21005-5055

DISPOSITION INSTRUCTIONS

Destroy this report in accordance with appropriate regulations when no longer needed. Do no return it to the originator.

DISCLAIMER

Information and data contained in this document are based on input available at the time of preparation. Because the results may be subject to change, this document should not be construed to represent the official position of the United States Army Material Command unless so stated.

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. This report may not be cited for purposes of advertisement.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF CONTENTS (CONTINUED)

LIST OF TABLES

FOREWORD

CONTENTS

This volume of the report on the AI Test Officer Support Tool investigation covers the application of neural computing techniques. This aspect of AI was examined to determine whether this emerging technology could contribute to future efforts of the investigation. Although this portion of the investigation represented less than 5 percent of overall efforts, both the nature of this work and the potential significance of the findings made it more appropriate to present the results in a separate volume, rather than in an appendix to the report. Consequently, this volume was patterned after TECOM Reg 70-12 format, with the notable exception that a separate methodology investigation proposal (usually appendix A) does not exist for this effort. The summary portion also pertains only to the examination of neural computing. Distribution will be in conjunction with volume I.

ACKNOWLEDGMENTS

The following personnel from Comarco, Inc. assisted in the preparation of this report under Contract Number DAEA18-87-C-0014:

Mr. Carlton Hall performed the research into neural computing and documented the findings in this report.

Ms. Linda Hulme, Ms. Gwen Gillen, and Ms. Genny Foster provided skillful assistance in the technical editing and word processing of the report.

The figures used herein are adapted with permission from Applying Neural Computing in Business, Government, and Industry, by Casimir Klimasauskas, NeuralWare Inc.

This page intentionally blank

SECTION 1. SUMMARY

1.1 BACKGROUND

A revival of interest in neural computing technology has developed over the last five years. The technology was originally developed over 40 years ago in an attempt to describe the functions of the human nervous system. After critical analysis revealed that the early models contained certain fundamental flaws, researchers in artifical intelligence (AI) largely abandoned this technical approach towards the end of the 1960s. Throughout the 1970s and early 1980s, researchers in the fields of neurology, psychology, mathematics, and physics continued to explore various aspects of models inspired by neurology and psychology. This research has led to several new and more complex models and has found solutions to some of the problems encountered with the earlier simpler models.

Since the start of the 1980s, research on neural computing has gathered new momentum. New models, combining several disciplines, have been proposed. The efficiency of these models for practical problem-solving activities, such as pattern recognition, has been demonstrated through software simulation. Recent technological progress in electronics and optics has made it possible to anticipate the appearance of economically viable hardware implementations. Over two hundred research organizations and industrial companies are currently active in this field. Several companies are producing software and hardware neural computing products for commercial application and for use by researchers and engineers to develop applications of the technology.

Projected Command, Control, Communications and Intelligence (C^3I) systems will include neural computer-based functions, [reference 1]. These projected C^3I systems will employ more traditional AI-based software and conventional embedded computer resources (ECR), as well as neural computer-based functions. These systems, with requirements to assure the security and survivability of intelligible, accurate, and properly identified data in the hostile environment of military operations, will be designed with distributed architectures and complex transmission protocols.

Testing these systems, with complex conventional ECRs, will stress the available test tools and challenge developers and users alike to supply tool capabilities which will keep pace with the test demands. As the projected C^3I systems with neural computer-based functions are implemented, the burden on the system tester will increase. This will require additional tools capable of testing neural computer-based hardware and software. Neural computing technology has the potential to supplement conventional methods and to help the tester and tools developer meet these testing demands.

1.2 OBJECTIVE

The objective of this investigation was to explore the architecture, concepts, and issues that must be addressed in applying neural computing to a particular problem. Specific goals were:

a. Define the building blocks of neural computing.

b. Present some of the basic neural computer architectures.

c. Define the issue of data preprocessing and representation.

d. Determine in general what neural computing can do today.

e. Explore application neural computers as adaptive expert systems for USAEPG.

f. Identify issues associated with testing of neural computer systems.

1.3 SUMMARY OF PROCEDURES

The initial approach was to investigate the field of neural computing for applicable methodologies and to relate them to particular areas of software and systems testing. This approach is particularly appropriate in dealing with the application of the developing discipline of neural computing.

1.4 SUMMARY OF RESULTS

Neural computer architectures are good at solving several kinds of tasks. Areas which have been identified that have varying degrees of practical impact today are modeling and forecasting, signal processing and pattern recognition, and pattern-classification expert systems.

a. In many problems, such as diagnostic systems or software maintainability assessment, the correlations between input variables may not be well understood. Neural computing provides methods of discovering the relationships between the input variables and how to use them in making a correct classification of the data.

b. A good problem for solution by neural computing has the characteristic of overlapping classifications such that one of two or more categories are probabilistic. Neural computer architectures are able to deduce these probabilities from training examples or historical data. This characteristic can be useful for knowledge acquisition from the examination of the results of an expert's activity in a specific domain. Some of the test planner AI system prototypes outlined in reference 2 could be developed using a neural computing solution. These prototypes include system subtest selection and global test resource estimating, where the total test time and the number of test items depend on the results of the subtest selection process.

c. One of the most attractive problems for solution by neural computers is an area in which the category boundaries of the domain knowledge change with time. The adaptability of neural computer architectures allows them to change the decision classification gradually as the conditions for making a particular classification change. The ability of an expert system to solve problems, justify recommendations, and document complex processes has made it a valuable tool. However, there are limitations to the traditional rule-based expert system, including:

(1) The labor-intensive nature of knowledge acquisition.

(2) The system's inability to learn or dynamically improve behavior.

(3) Unpredictable behavior when operating outside the specific design of the knowledge base.

d. Neural computing provides potential solutions to these problems. The capabilities include:

(1) Training from examples of expert actions.

(2) The ability to adjust dynamically to changes in the environment.

(3) The ability to generalize from specific examples.

(4) Tolerance to noisy or random inputs.

(5) Graceful degradation in problems outside the specific range of experience.

(6) The ability to discover complex relationships among input variables.

1.5 ANALYSIS

The representation of the input data and the preprocessing of the data, either by conventional methods or by the neural computer architecture, is a key element in the development of a neural computer application. In building an application, the problem must be broken down into basic steps. These steps are very similar to the analytic process used in the development of a conventional digital computing application. They are as follows:

a. Selecting the recognition problem and collecting sample data.

b. Developing feature extraction and encoding methods.

c. Designing a system architecture.

d. Training and testing with sample data.

e. Evaluating results.

f. Revising the feature extraction method and architecture to improve the results.

g. Merging the recognition package with a user interface to complete the application.

With the proper selection of the neural computer system architecture, some of the steps can be performed through "auto-association" and "self-organization" of the dynamics of the neural computer system. Neural computing does not remove the need for domain experts and knowledge engineers. Instead, the knowledge acquisition process is simplified. The domain expert is necessary to define the relevant data. The knowledge engineer is required to determin how the data should be preprocessed and how to obtain the best possible performance from the neural computer system.

Neural computers may function as stand-alone expert systems, or in conjunction with rule-based systems. Stand-alone neural computer expert systems have difficulty explaining their behavior and are unsuitable for problems that do not involved classification. The most promising approach appears to be a hybrid of neural and rule-based expert systems. In a hybrid expert system, the neural computer can act as a preprocessor to perform pattern recognition or sensor analysis. The neural computer can also perform as an internal component for learning, generalization, or classification. Combining neural and rule-based approaches to expert systems may appear to be a research project, but it has been successfully applied to manufacturing operations.

1.6 CONCLUSIONS

Neural computing does not produce exact results in the same sense as the very precise computations performed by traditional digital computers. Rather, neural computing produces highly organized conceptual classifications, using pattern matching concepts. This capability can be utilized effectively in expert systems for knowledge acquisition and representation, through the use of pattern-based rules. Pattern-based rules address some of the defects of knowledge-based expert systems [reference 3]. There can be improvements in the areas of adaptability and generalization [reference 4]. However, neural computing is not a panacea for solving all of the problems of current expert system architectures. A neural computer architecture must be constructed and used with a clear understanding of the limitations and unique testing requirements. At the current stage of the technology, there are a few things which neural computing does better than any other technology. As the number of engineers who understand the technology grows, their individual incremental contribution will add up to a revolution [references 1, 5].

1.7 RECOMMENDATIONS

Neural computer technology has potential to enhance the efforts to achieve the goals for application of AI within USAEPG.

As a result of this investigation of neural computing technology, the following recommendations are made:

a. A collection of fundamental papers and books aimed at classroom and educational use should be assembled for a basic bookshelf on neural computing.

b. PC-based neural computer software tools should be investigated to support neural computer network architecture design and simulation.

c. A presentation should be prepared to demonstrate the basic concepts and applications of neural computing. This presentation should augment this neural computing investigation report.

d. An investigation of neural computer-based systems entering the military environment should be initiated to identify testing needs not yet addressed.

e. An investigation should be initiated to define testing procedures required for systems with embedded neural computer architectures.

f. A prototype of an expert system, with an embedded neural computer architecture, should be developed to demonstrate the application of "decision making by pattern classification" and knowledge acquisition from training examples or historical data. This prototype should be a test resource estimator, because of the availability of historical data to train the classification network model.

This page intentionally blank

SECTION 2. DETAILS OF INVESTIGATION

The history of neural computing dates back to the early 1950s and has paralleled, in the shadows, the development of AI. Researchers over the last 40 years have worked to instantiate these adaptive, associative models of biological systems as procedures for handling real-world information processing and computation [reference 6].

Thousands of scientists, engineers, and students are studying, developing, or applying neural computing models. They cover both biological models of brain behavior and technological models for implementation in government and industrial applications. Many engineers have been drawn to the field because neural computing researchers have discovered promising approaches to problems that require adaptive, massively parallel, fault-tolerant solutions.

A neural computer is trained to model the relationship between input and output variables. This model is then used to forecast, from a set of input variables, the expected output from the actual system or process. One example of this is attempting to predict the high/low temperatures and the likelihood of rain for the next three days, based on what has happened during the past three days. In this case, the inputs are a moving window of the temperature and humidity over the past three days, and the output is the predicted temperature and probability of rain, projected for the next three days.

Neural computer networks have demonstrated many capabilities, including fault tolerance, retrieval of nearest-neighbor data in pattern matching classification, self-organization and adaptive learning capabilities utilizing input training sets, associative recall to perform pattern patching on partial or degraded inputs, discovery of significant statistic features in data, and the ability to solve problems subject to combinatorial explosion under traditional algorithmic techniques.

Neural computers will run in real time when the architectures are implemented compactly in specialized hardware. The most advanced neural computer architectures provide intelligent systems capable of autonomous learning and skillful performance in complex and noisy environments. Such examples and future possibilities have generated a high level of enthusiasm among people working in the field. Much of this enthusiasm comes from the fact that many neural computer design principles, mechanisms, and architectures were discovered through analysis of the human mind and its neural mechanisms.

2.1 NEURAL COMPUTING APPLICATION

Neural computing research has demonstrated the effectiveness of these specialized software or hardware systems for pattern matching of noisy, distorted, and partial input patterns, and for associative distributed memory, as well as large-scale multiclass discrimination. The first multiprocessor boards, operating as coprocessors on digital mini- or microcomputers, are now available, and some chip sets will be available in the future. As cost declines, these non-algorithmic processing units will offer an alternative for efficient and effective processing of noisy, incomplete, or inaccurate data.

Most of the work to date has been done in academic research institutes, but major advances have prompted several major companies, such as AT&T, Bendix, Texas Instruments, TRW, and General Electric, to develop in-house neural

computer projects. Commercial products are now being offered to industry and government, including the Department of Defense, by start-up companies such as Hecht-Nielsen Neurocomputer, Synaptics, NeuralTech, NeuralWare, and Nestor.

Conferences, such as the IEEE First Annual Conference on Neural Networks (June 1987) and the American Institute of Physics Conference on Neural Networks for Computing (April 1986), have emphasized the applied capabilities of the technology. Projects have included adaptive filtering and signal processing, conversion of printed words to spoken speech, handwriting recognition, letter and word recognition, identification of target vehicles by radar or sonar patterns, robotic and control applications, various vision processing applications, financial and economic forecasting, data compression, and decision making as pattern classification. Neural computers can self-learn relevant patterns in data. When presented with a statistically valid sample, these trained systems then have the capability to store large numbers of feature vector patterns in the distributed network memory, and to perform nearest-neighbor associative pattern matching from noisy, degraded, or partial input patterns.

The identified characteristics of neural computing that have practical application impact today are in three major areas:

a. Modeling and forecasting.

b. Signal processing.

c. Pattern-classification expert systems.

2.1.1 Modeling and Forecasting. In modeling a system or process, the objective is to find some defined relationship between several input attributes and one or more resultant output conditions. Once the relationship based on a number of sample data points has been found, it is used to interpolate between or extrapolate beyond the sample data.

Certain types of neural computer architectures are very good at synthesizing this type of relationship. Experiments have shown that a neural network performs very nonlinear, least-mean-square regressions to develop a continuous mapping from one multidimensional space to another. These experiments have further shown that for a fairly complex test case, a polynomial regression technique (Group Method Data Handling) was accurate to ±13 percent, while the neural computer was accurate to ±5 percent [reference 7].

This neural computer-based modeling and forecasting approach has been applied to very difficult problems for servo control, sensor processing, structural design, and computer-aided design/computer-aided manufacturing (CAD/CAM) modeling. In the CAD/CAM field these techniques have been used to perform computer-aided analysis and testing. After a mechanical or electronic system has been designed, its probable performance can be predicted in two ways: by actual physical testing or by computer analysis. The latter is attractive because it is quicker, easier, and cheaper than physical testing. After the synthesized model has been validated, it can be used to predict the system's probable performance under a wide variety of conditions.

2.1.2 Signal Processing. The field of adaptive signal processing grew out of work with very early neural computer models. Neural computer networks have evolved which perform better than other techniques for noise filtering, matched filters, and data compression.

One example of this is a neural computer network designed to form a Fourier transform and produce a digital feature vector which characterizes, in a highly compressed form, the original input image. This type of network provides two major advantages over more traditional programming techniques. One advantage is the response time to produce a solution. The other is the relative ease of implementation of the image recognition solution. As different types of images are presented for analysis, the network can "learn" to identify the unique relationships of the different patterns. Thus, when an unknown image sample is presented, it is processed against the known patterns to determine the degree of similarity or to define a new classification.

2.1.3 Pattern-classification Expert Systems. Expert systems are an important element applicable to efforts in achieving the goals for the application of AI technology to the needs of the C^3I software and systems testing community within USAEPG. The principal power of expert systems is the embedded knowledge base. As the development and application of expert systems increases, the problem of how to acquire knowledge must be addressed. The standard process of knowledge acquisition has been performed by a knowledge engineer, who interviews an expert, identifies the structure of the expert knowledge, and builds rules for the expert system. This task is complicated by the fact that the experts generally have not analyzed their thoughts, are not explicitly aware of the structure of their knowledge, and cannot provide an overall account of how decisions are made. Thus, knowledge acquisition has accounted for a major part of the overall development time and expense of most expert systems. Therefore, the problem is not only how to acquire knowledge, but how to automate the acquisition of knowledge.

The characteristics of neural computing technology have a high potential for practical application to the problem of expert system knowledge acquisition and representation. Through the use of pattern-based rules, some of the defects of rule-based expert systems, such as adaptability and generalization [reference 3] are addressed.

Many of the decisions an expert makes each day are really pattern-classification problems. Given a set of facts, conditions, or attributes, questions such as the following could be answered: Should this individual be hired or promoted? Should a particular test be applied to this system? Is a given resource needed to test this function?

These and many other questions draw upon the ability of an expert to classify, consciously and unconsciously, a pattern of perceived facts into one of several possible categories. When these classifications lead to action, this is called "decision making by pattern classification" [reference 7].

The traditional expert systems usually examine a single feature or aspect of a situation at a time, and, based on a defined set of rules, attempt to narrow the possible outcomes step by step. Neural computer-based pattern-classification expert systems derive an understanding of how to classify a particular set of inputs by looking at self-organized prototype classifications. Thus, the pattern-classification expert system becomes a trained model

of the expert, with a self-organized relationship between the input attributes and the resultant output conditions or actions. However, this trained model must be tested and analyzed to determine that the relationships are producing the correct action for the right reason. Verification testing must determine that the network has learned valid relationships.

The simplest kind of classification problem is to identify some unknown object or phenomenon as a member of a known class of objects. These classes are prototypes that are hierarchically organized, and the process of identification is one of matching observations of an unknown entity against features of known classes. An example is identification of a plant or animal, using a guidebook of features such as coloration, structure, and size.

Figure 2-1 illustrates this classification process. The expert system classifies readers' personalities and selects books that they might like to read. Two classification problems are solved in this example. However,

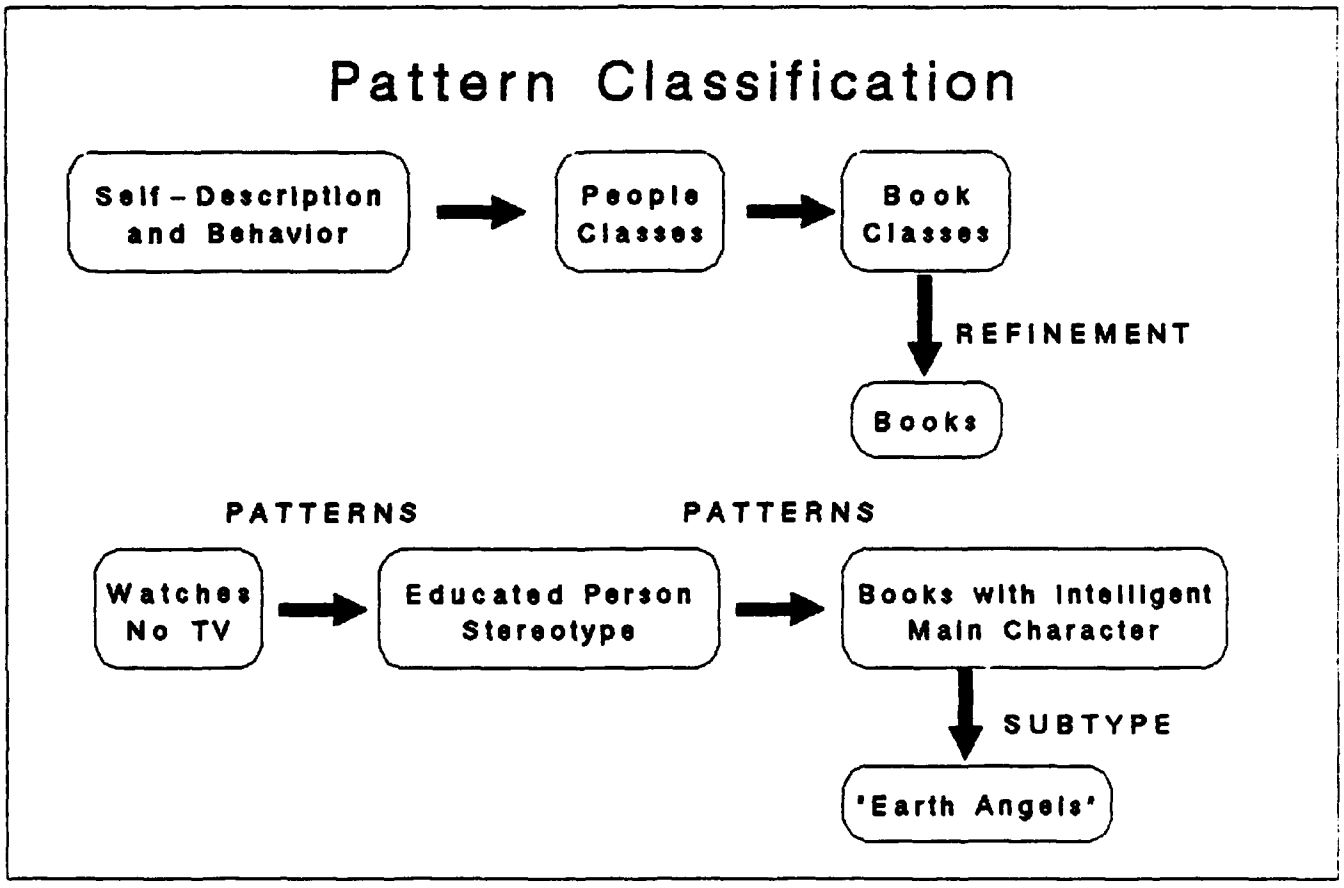


Figure 2-1. Inference Structure of Pattern Classification

people and book classifications are not distinct. For example, "fast plots" is a characteristic of books, but "likes fast plots" is associated with a person prototype. The relation between a person prototype and "fast plots" should be distinguished from abstractions of people and books. One objective of the system is to learn better people prototypes (user models). The classification description of the user modeling network shows that it should also be learning better ways to classify books. The network does not need to perfect the user model before recommending a book. Refinement of the person prototype occurs when the reader rejects book suggestions.

If knowledge acquisition is viewed as the transfer of problem solving capability from some knowledge source to an expert system, then a knowledge source may be either a direct source, such as a human expert, or an indirect source, such as data and examples of how a task is performed. Thus, knowledge for use by an expert system may be obtained from a human expert, from data and examples, or both.

There are three approaches to knowledge acquisition: interviewing, learning by interaction, and learning by induction. Interviewing and learning by interaction are very similar, simply involving different degrees of interaction with a knowledge engineer. However, in both cases the expert is the principal source of the knowledge.

In learning by induction, the role of both the knowledge engineer and the expert are diminished by a significant degree. Learning by induction focuses on algorithms that analyze data and examples and generalizes from them to obtain knowledge. The main problem then becomes identifying the suitable characteristics or attributes on which induction should be performed.

Neural computing technology can provide a significant improvement in areas of learning by induction. This improvement can be applied to reduce the cost of knowledge acquisition, through automation of the process, and also, because studies show very weak correlations between verbal reports and mental behavior of humans, it has the potential to improve the quality of the resulting knowledge base. With an embedded neural computing element, learning by interaction and learning by induction can be combined in a uniform framework for knowledge acquisition. The key point is that the weaknesses of each approach to knowledge acquisition can be compensated for by the inherent strengths of the other.

The basic idea is to have an expert system that learns to perform a task from examples of an expert's actions. To use this learning system for knowledge acquisition, an expert provides a set of examples of different types of decisions, and a data base containing the relevant features or attributes which influence the decisions. The learning system then analyzes the example data to identify the important constructs and criteria used in decision making, and discovers or forms a model of the knowledge that directs the "mental behavior" of the expert system.

When the expert system is presented with a new set of input data, two possibilities exist: either the same set of data has already been classified by the system, or the data presents a new case that does not exactly match any of the classifications provided by the previous examples. In the second case the expert system must "guess" which one of the classifications the expert would have made in the new case. Some of the "guesses" or classification

assignments are much better than others. Therefore, the embedded neural computing classification could assign a "confidence factor" to each of the possible classifications, and perhaps provide a justification or a view of how the classification was made. The user of the expert system would make the final classification decision, or if the system is being trained, the expert training the system could make the final classification decision or assign a new classification.

The adaptability of neural computer networks allows them to change as the conditions for making a particular classification change. One example of this is attempting to predict the likelihood of rain in various parts of the United States, using a neural computer network trained on data collected in Arizona during the month of May. If this system were used to predict rain in Arizona in the month of July, or in Kansas in the month of May, the network would likely need additional training to reflect the change of location and time. The neural computer architecture remains the same, but the knowledge within the network is adapted to the new conditions through training.

2.2 NEURAL COMPUTER CONCEPTS

Neural computer networks are highly distributed, non-algorithmic computation systems based on multiprocessor architectures and dense interconnection schemes, which may be instantiated in hardware or run by software or hardware simulators.

The basic components of neural computers are analogous to the components of the conventional digital computer systems, which are constructed from resistors, diodes, and transistors, combined in large numbers and in a variety of ways. Thus, combining large numbers of the basic components of neural computing in a variety of ways leads to large neural computing systems with many interesting and practical properties.

2.2.1 Basic Components of Neural Computers. The basic components of the neural computer are the processing element (PE) and the interconnect or connection, as shown in figure 2-2. The connection has an associated connection strength. Input signals enter the PE through a connection which regulates the signal via the connection strength [reference 5].

The PE performs two basic operations, the summation function and the transfer function. Signals that enter the PE through the weighted connections are summed by a summation function, as shown in figure 2-3. This summation is then transformed by the PE transfer function, which creates the net output that is a function of the weighted sum of all the input signals. An output signal from the PE is created (PE is activated), if conditions (activation level) within the transfer function are met. If the activation level is not met, an output from the PE is not produced (PE is not activated). Transfer functions may be either linear, non-linear, or step functions [reference 7].

Each neuron-like PE has a small amount of local memory and can perform its entire computation function based only on local data passed to it via interconnects with other PEs. Transfer functions, which are modified discrete integral products of weighted sums, define the computation performed in each PE.

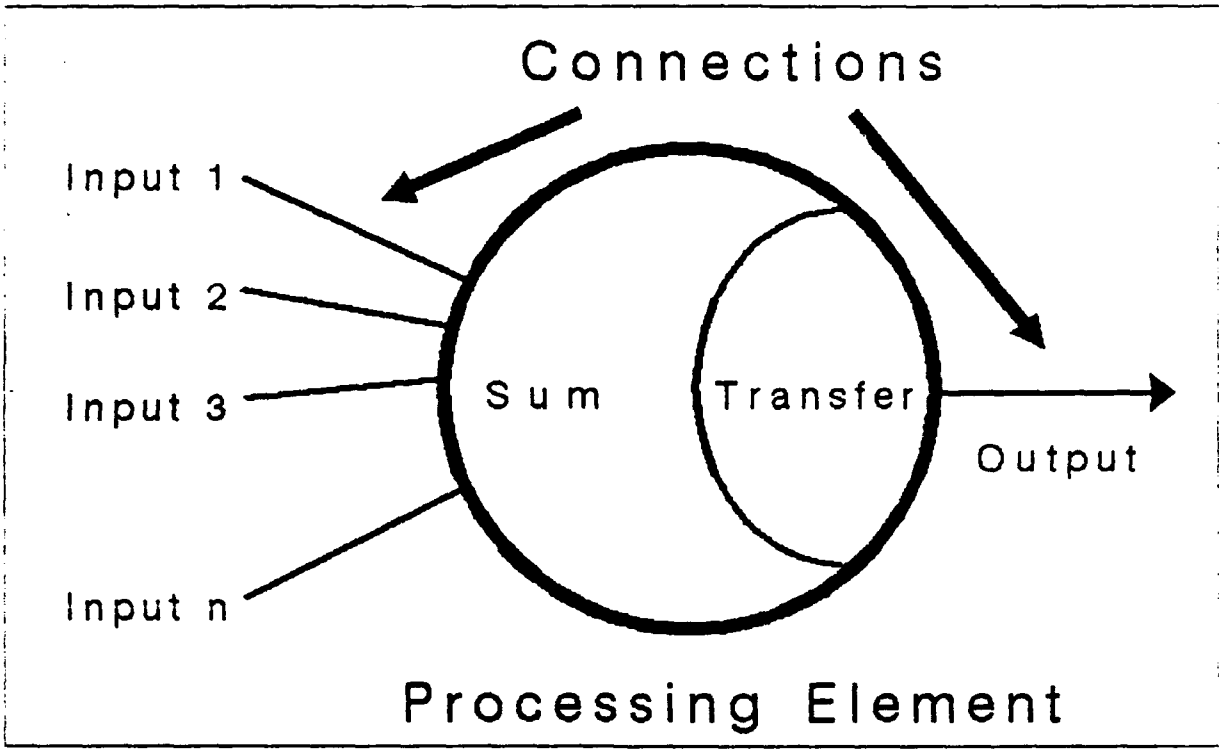


Figure 2-2. Basic Components of Neural Computers: PE and Connection

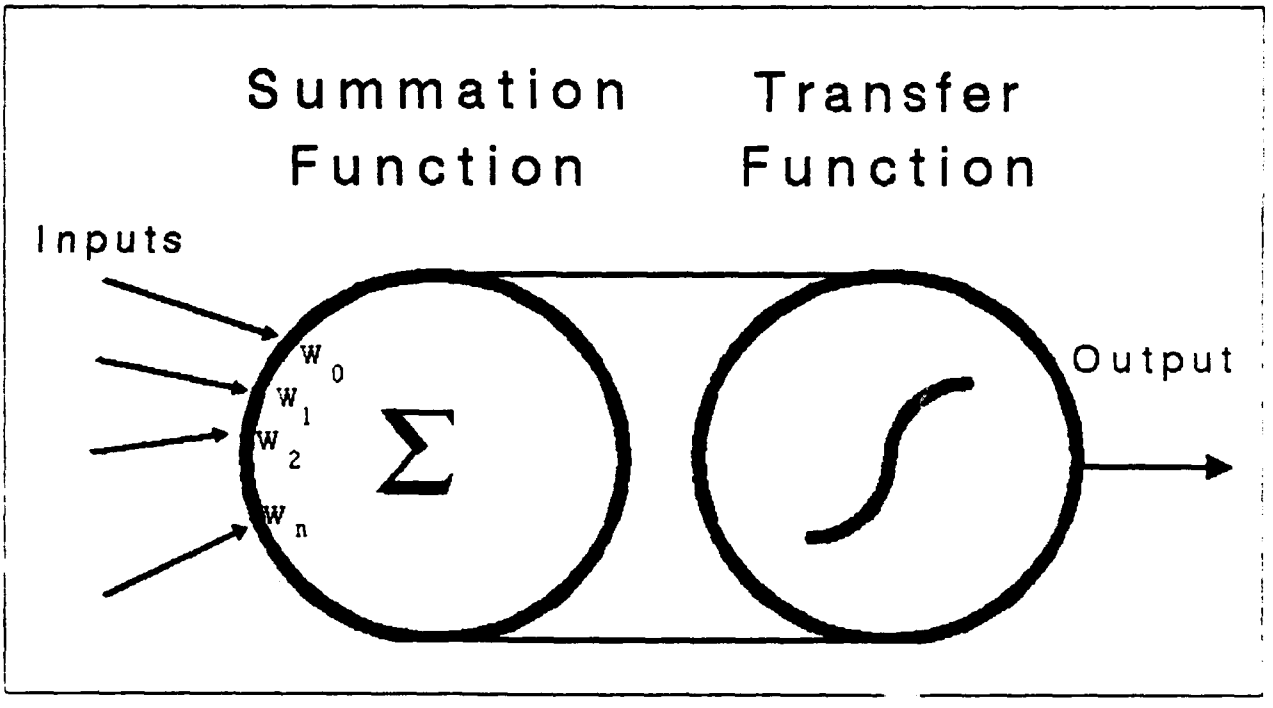
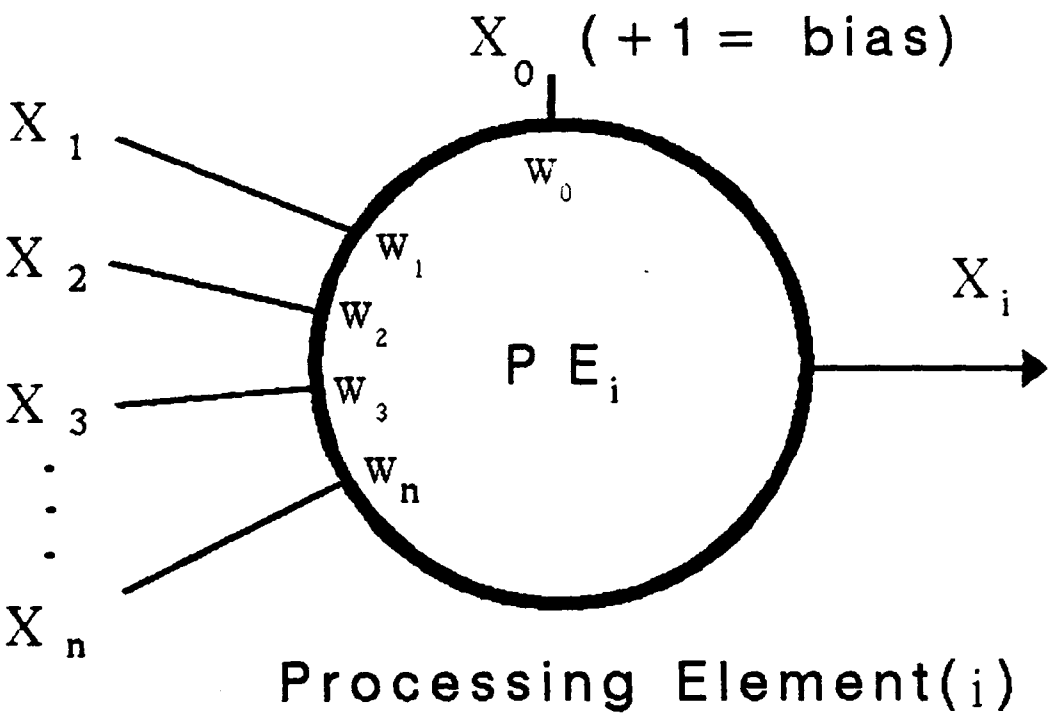


Figure 2-3. Processing Element Model

Figure 2-4 shows a mathematical representation of a PE and its connections. This representation describes the "neurodynamics" of a PE. A sigmoid transfer function is shown in the representation. There are many variations on these basic components.



Neurodynamics:

Summation Function:

$$I = W_0 * X_0 + W_1 * X_1 + W_2 * X_2 + \ldots W_n * X_n$$

Transfer Function:

$$F(k) = (1 + e^{-k})^{-1}$$

Output:

$$X_i = F(I)$$

Figure 2-4. Mathematical Representation of a Processing Element and Its Connections

2.2.2 Neural Computer Networks. When a number of PEs are connected together into an array which allows them to communicate, the result is a neural network. Within a neural network, PEs are grouped together to form layers, which simplifies the description of the network operations. Thus, one or more PEs grouped together, with the same neurodynamics, is a layer [reference 5]. A subgroup of a layer which contains a logical group of PEs is termed a slice.

Figure 2-5 shows a neural computer network consisting of three layers. Data is applied to the input layer. Connections transfer information from the input layer to the hidden layer, and from the hidden layer to the output layer. The middle layer is called a hidden layer because its inputs and outputs are not available outside the network.

Each PE has a single output. Input to one PE from another PE has a weight or adaptive coefficient which acts as either an inhibitory or an excitatory influence on the input. Interconnection schemes vary from fully connected networks, where every PE is connected to every other PE, to layers where the output of each PE on one layer fans out as input to each PE on the next layer. In figure 2-5, all of the connections are from a previous layer to the next layer, which forms a feed-forward network.

Nonlinearity may be introduced to the network by allowing PEs to feed back the outputs of a layer, or group of layers, to their inputs. This is called a feedback or resonant network. Nonlinearity is a key feature that differentiates the neural computer from other, more traditional, algorithmic processing techniques such as polynomial curve fitting or factor analysis.

Within a layer the PEs may be required to "compete" to provide the single or a partial output of the layer. When competition is introduced, the model of the PE is extended as shown in figure 2-6.

2.2.3 Learning and Adaptation. Neural computer networks are used to develop a relationship between the network inputs and outputs. In a conventional digital computer system, this relationship is provided by the software algorithmic program, developed from a functional specification. Expert systems use concepts or logic, described in a knowledge base, to develop an "inferred" relationship.

Neural computer networks are not given information about how to process the input data to produce the desired output. The network performs this function through the use of a "learning rule", which adjusts the connection weights in the PEs in such a way that the desired output is achieved. This ability to determine the processing function is called self-organization, and is also referred to as learning or adaptation.

Figure 2-7 shows the general learning process [reference 7]. The network starts with the connection weights in the PEs set to random values. Pairs of inputs and outputs are applied to the network. These pairs of data are called the training set or historical training data. Based on the input and the expected output pair, the network learns by adjusting the connection strengths between PEs. The method of performing the adjustment or adaptation is called the learning rule. Depending on the learning rule, a network may require several thousand presentations of the training data before it converges to the required configuration.

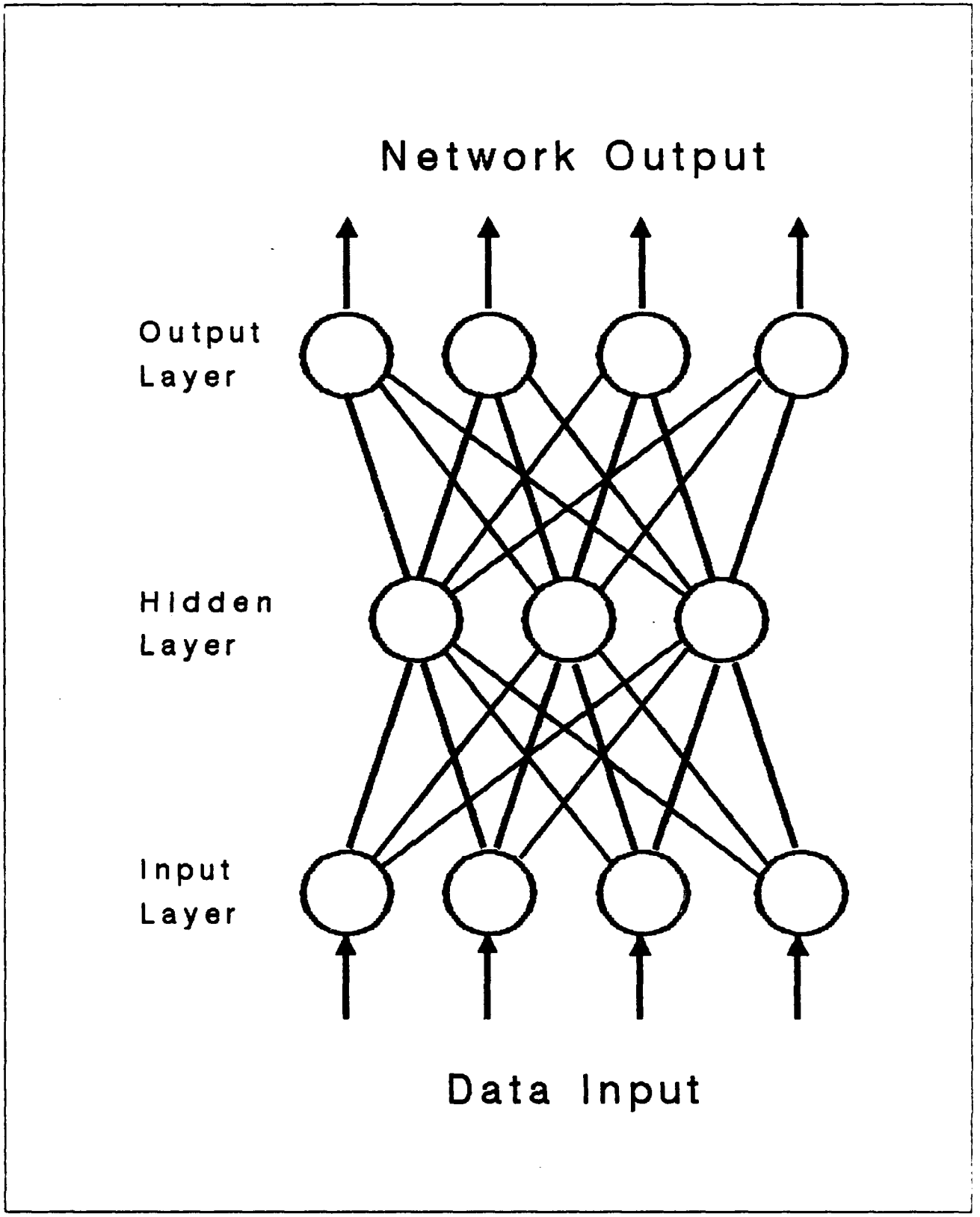


Figure 2-5. A Three-layer Neural Network Architecture

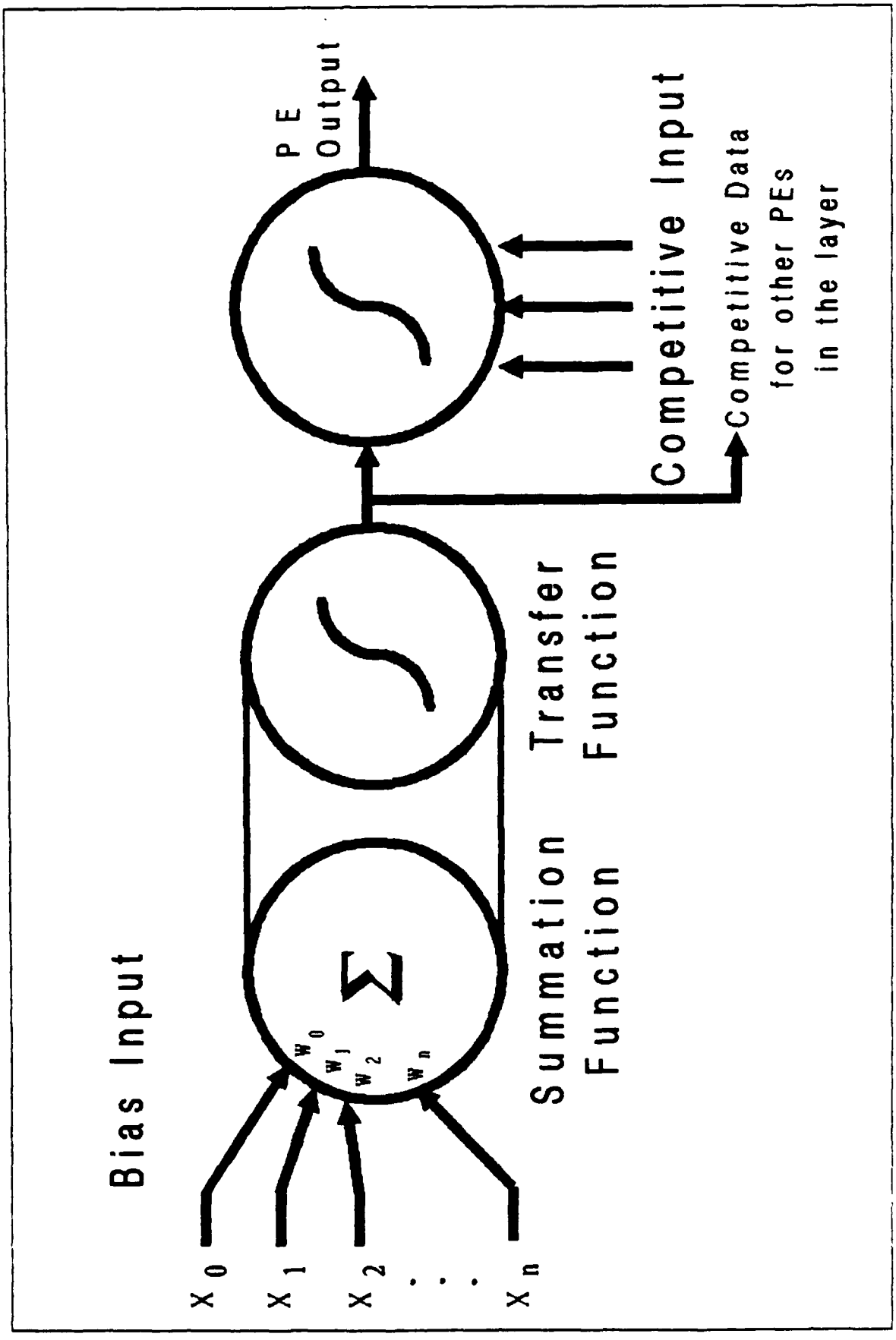


Figure 2-6. Competing PE Model

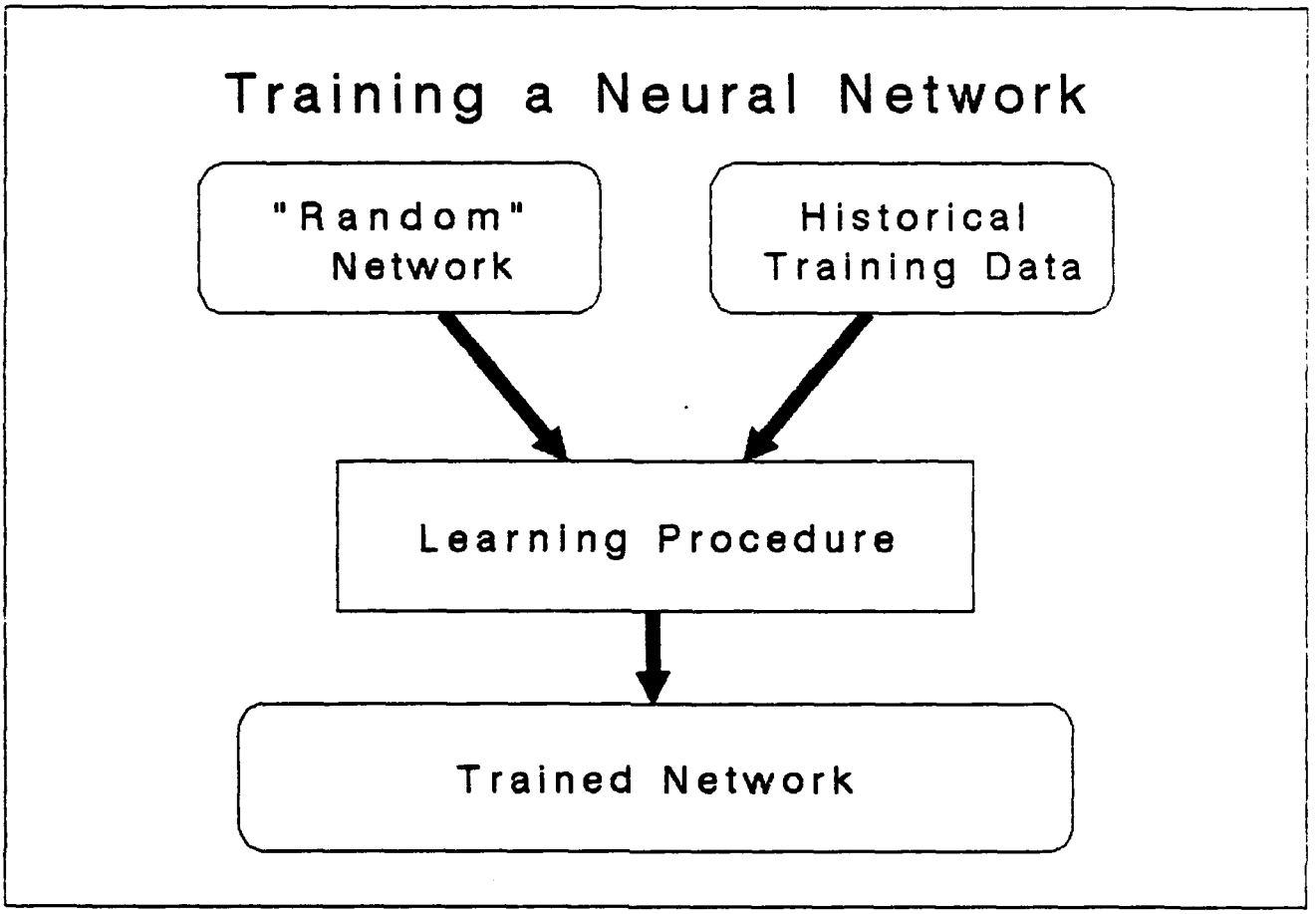


Figure 2-7. Neural Networks Learn from Historical Training Data

When only the input data is presented and the network must organize the data into related groups or classifications, the learning process is called auto-association or unsupervised or self-supervised learning.

2.2.4 Network Architecture. The organization of the network is called the network architecture. The network architecture is designed to perform a general classification task. The learning process occurs in the context of the network architecture. This architecture is constructed based on four components:

1. The number of PEs.
2. The configuration of the connections and layers.
3. The neurodynamics of the PEs.
4. The network or layer learning rules.

The number of PEs and the configuration of the connections and layers determine what degree of detail the network can organize and process.

Neurodynamics and connection strategies affect the way in which a network processes data. Connections can transfer information in a single direction in

a feedforward network, or in both directions when feedback is provided, to produce nonlinear operation during learning and recall. These factors determine the operating network dynamics.

Several neural computer network architectures have been defined and are considered to be "standard" architectures. Connection strategies, transfer functions, learning rules, and other variables follow guidelines developed through neural computer engineering and research. As neural computing applications are developed, the standard architectures are modified and combined with other architectures to achieve the desired effect and accomplish the goals of the application.

2.2.5 Data Preprocessing and Representation. The representation and preprocessing of the input data, either by conventional methods or by neural computing methods, is a key element in the development of a neural computing application architecture. As with any application, the proper formulation of the problem, the representation of the knowledge or data, and the presentation to the system for solution will affect the process or strategy taken to solve the problem.

Some of the representation issues that must be considered are as follows:

a. It is important that all of the pertinent factors are represented in the data that is presented for input.

b. It is sometimes useful to preprocess the raw input data by integrating parameters in a way that separates pertinent factors.

c. Research seems to indicate that the best representational division of data is the one that the problem domain expert would use. Thus, if the domain expert usually forms the ratio of two numbers in the solution process, that ratio should be used as the representation of the data presented to the input for a neural computing solution.

Most architectures require that input data be preprocessed for normalization. There is also a need for scaling the data prior to normalization. Scaling spreads the data so that one factor does not dominate another. Scaling also centers the data, which is necessary for normalization. Scaling and centralizing data usually involves forming one or more statistical parameters for the input training set.

2.3 STANDARD ARCHITECTURES

With the great variety of possible neural computer architectures, a review of even the major ones would be beyond the scope of this report. A brief description of the interesting properties of two neural computer architectures will illustrate some of the basic design characteristics.

2.3.1 Perceptron Architecture. The perceptron neural computer architecture was designed by Frank Rosenblatt [reference 6] to model the pattern recognition capabilities of the human visual system. The architecture is relatively simple and illustrates several of the basic ideas related to PEs. It is a feed-forward network, without feedback or cross talk between PEs.

The network consists of three layers. The input layer is a fan-out buffer that presents each input value to each PE in the next layer. The second layer contains a set of hard-wired feature detectors to detect specific features from the input pattern. The output layer contains perceptrons or adalines (adaptive linear neurons), which act as feature recognizers. The weights on the connections to the input and detector layers are all fixed, while the connection weights on the output layer are set by the learning procedure to train the PE to recognize and identify a set of patterns. Figure 2-8 shows a perceptron network [reference 5].

Each perceptron layer PE has one input tied to a constant value of plus one through an adjustable weight. The other inputs are connected through adjustable weights to the output of the PEs in the feature detector layer. The output transfer function is zero if the weighted sum of the inputs is less than or equal to zero. Otherwise, the output is equal to the weighted sum of the inputs, or alternatively, the transfer function can be configured to provide an output value of one. In the latter case, the PE is termed a binary or threshold logic unit.

The basic learning rules for training the perceptron weights are as follows:

1. If the output value is correct, then do not change the weights.

2. If the output is zero and should be greater than zero, then increment the weights on the "active" input connections.

3. If the output is greater than zero and should be equal to zero, then decrement the weights on the "active" input connections.

An input connection is "active" if it is greater than zero.

A perceptron system is guaranteed to find a set of connection weights that correctly classifies the input vectors, if such a set of weights exists. Since the perceptron learning procedure can be applied independently to each of the output units, it will find the mapping from the input vector space onto a classified set of output vectors, if such a mapping classification exists. The problem is that unless the classes can be separated by a line or a plane, with all of the first classification on one side and all of the second classification on the other side, the mapping does not exist. All functions or vector spaces for which such a plane exists are called linearly separable. Thus, a perceptron can linearly separate a set of inputs, but it cannot do more [reference 5].

There are a number of problems which can be solved with units of this type. However, most real problems cannot be solved without significant problem-dependent preprocessing to form a linearly separable data set. The perceptron, therefore, is interesting as a device for obtaining an understanding of concepts, but it is not generally useful for applications.

Providing solutions to the linear separability and other problems of the perceptron are some of the recent major advances in neural computing.

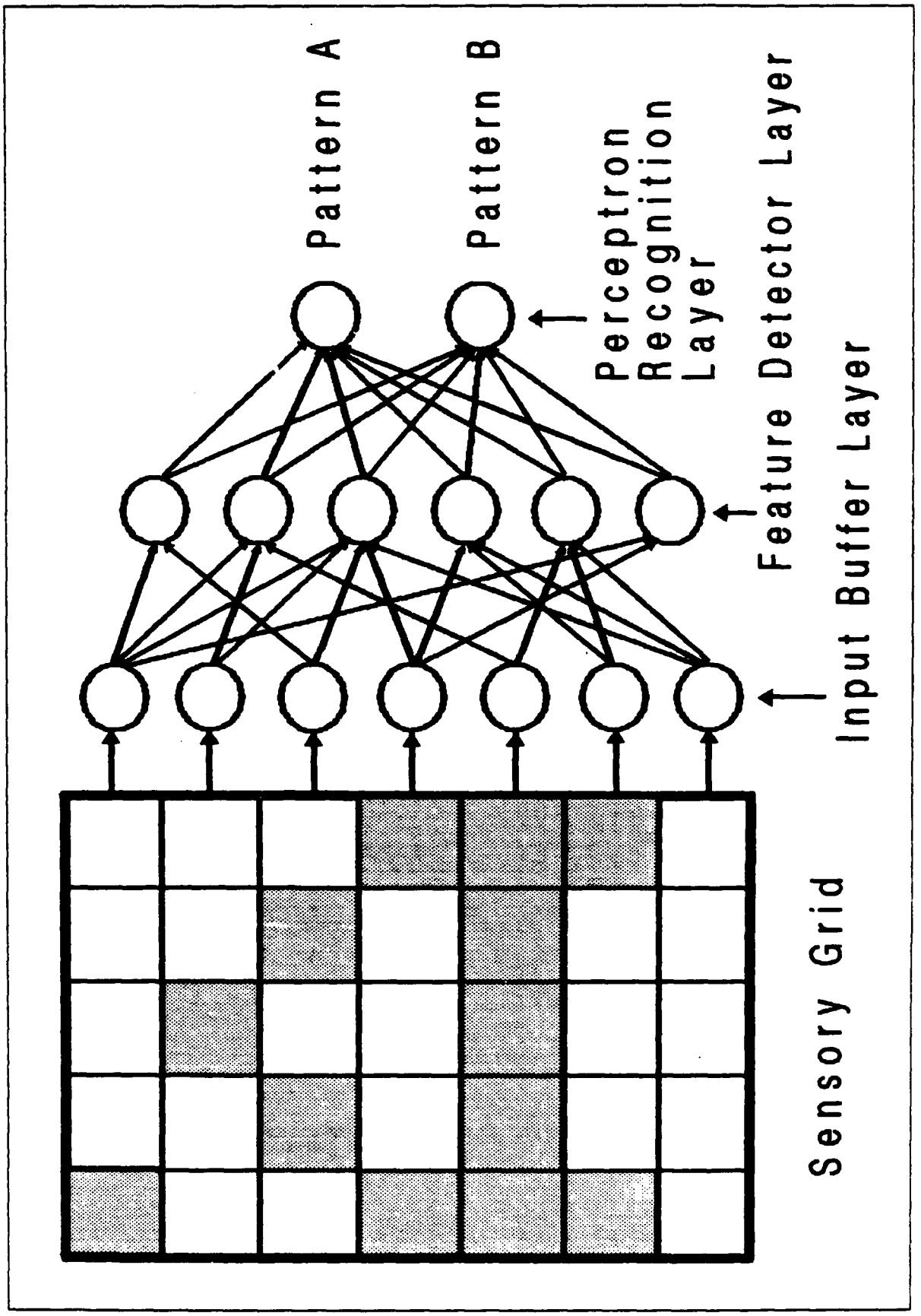


Figure 2-8. The Perceptron Network

2.3.2 The BPN and CPN Architectures. The "backpropagation" (BPN) and "counterpropagation" (CPN) network architectures are representative of configurations that can be effectively applied to the development of a large range of applications. Both are mapping functions, which map n-dimensional input vectors to m-dimensional output vectors. BPN, developed by Rumelhart [reference 8], is one of the more widely known neural computing architectures, due to its extensive use by members of the Parallel Distributed Processing or "PDP" group [reference 8]. BPN is a generalized gradient descent algorithm, which adaptively adjusts its connection weights during training to minimize the mean squared error between the actual and the desired output. Once the output of a computation reaches the final output layer, the error values are sequentially propagated to the previous network layers.

The CPN architecture was developed by Hecht-Nielson, based on work by Kohonen [reference 9]. The CPN trains itself to be an optimal, equiprobable lookup table. In its complete form, CPN performs in both a forward and a backward mode.

Both architectures require a period of "training" before test data is submitted for classification. A statistically significant set of training data must be obtained. The training set consists of (X,Y) pairs, where each X is the input feature vector and the corresponding Y is the desired output vector for the mapping. The training set is repeatedly submitted to the network until it converges or is fully trained. Once the network has been trained, the adaptive learning function is turned off and feature vectors are presented for testing. The network produces an X-to-Y mapping based on its "learned" structure.

The internal structure, or "memory", is based on a set of weights associated with each PE (see figure 2-4). The PE has several inputs, each with an associated weight and a single output. The transfer function for the PE varies according to the layer, and is based on a discreet integral of the product of each X and its associated weight. During the training phase, a learning function describes the adjustment of the PE weights for each layer and network type.

BPN encodes its memory as a set of internally determined features on the hidden layer of the network. The set of weights across all PEs on this layer represents a distributed memory. After adaptive adjustment, or training of the weight according to the least mean criteria, an input feature pattern stimulates a distributed pattern of activity in the hidden layer, which is interpreted by the output layer as a particular class of pattern.

The hidden layer of the CPN acts as a set of matched filters, which generalize the input patterns during training. During testing, the resultant mapping function finds the nearest match, from among the generalized PE exemplars, to the input pattern. CPN is attractive because the hidden layer PEs can be viewed as a generalized statistical distribution of the features. This property could be used to statistically track drift in the input patterns.

2.4 THE NEURAL COMPUTER WORKSTATION

The software or hardware instantiated neural computer workstations which are currently available consist of a personal computer (PC) host with a color monitor, either a hardware-based coprocessor neural-computer-network simulator or a software-based neural-computer-network simulator, a user interface library, and a set of prepackaged network elements, including summation functions, transfer functions, output functions, and learning rules.

Neural computing application programs are written in a high-level language such as C on the host computer. The application program is responsible for preprocessing the input data patterns, initiating calls to the hardware/software network simulator via an interface-procedure library, and obtaining the network results from the simulator for final analysis and output.

A typical coprocessor network simulator board is capable of implementing up to 30,000 PEs and 480,000 PE interconnections. Interconnections are updated at a rate of 25,000 per second during learning and 45,000 in the recall mode. The cost for this hardware is $7,000 to $15,000.

A typical software network simulator is capable of implementing up to 4,000 PEs, and 15,000 PE interconnections with 640 KB of memory or up to 425,000 interconnections with 8 MB of extended memory. The interconnections are updated at a rate of 8,000 per second without an 80287 math coprocessor, and 32,000 per second with the math coprocessor on a 10-MHz PC. The cost for this software is $500 to $1,000.

Both the hardware and software neural computing simulator systems use disk files to store the network configuration such as network constants, which define the specifications (number of PEs, layers, input/output elements, etc.), the weights for all PEs in the network, and the output state information for all the network layers. These files completely characterize the current state of a network configuration.

A neural computer network simulator has successfully discriminated between sonar returns of an undersea rock and a cylinder. The network consisted of less than 100 PEs, with approximately 2,000 interconnections. This network accurately distinguished between rocks and cylinders 90 percent of the time, which compares favorably with the performance of human operators [reference 1]. Therefore, these hardware/software neural computer network simulators are capable of performing the types of practical applications outlined in this report.

2.5 CANDIDATE PROTOTYPE APPLICATIONS

Application areas and specific application projects have been identified through discussions with USAEPG testers, test officers, and managers. The development of these prototypes will provide some of the tools described in the development of the test planner and test player prototypes outlined in reference 2.

The potential applications of neural computing technology listed in table 2-I are primarily as embedded modules in expert systems for "decision making by pattern classification" and as knowledge acquisition from training examples

TESTING APPLICATIONS / TECHNOLOGY	Software Static Analysis	Test Planning	Measurement Design and Analysis		Projection of Test Requirements and Results
			Scenario Design & Building	Data Reduction & Analysis	
Expert Systems	X	X	X	X	X
Planning		X			
Scheduling		X	X		
Natural Language		X	X		
Knowledge Representation	X	X	X	X	X
Neural Computing	X	X	X	X	X

Table 2-I. Technology Applications Matrix

or historical data. Neural computing technology can also have application to modality propagation [reference 2] as an aid in the application of statistical methods in conjunction with modal systems that are embodied in many expert system inference engines. A good problem for application of neural computing methods has the characteristic of overlapping classifications such that one of two or more categories is probabilistic. The strength of the classification assignments can be used to deduce a numeric measure of belief, certainty, likelihood, etc., of a classification assignment.

2.5.1 Preprocessing Statistical Data. Much of the data produced during C³I system testing requires statistical processing and analysis to derive meaningful performance information. The suitability of the collected data for analysis is determined, to some extent, by the statistical design of the test, which makes explicit the assumptions and constrains the type and volume of data to be collected, to assure that the desired analysis can be performed.

In the course of a given test, many situations arise which can lead to variation in both the plan and its execution. In such situations it may be difficult or impossible to determine whether the gathered data meets the original statistical requirements, or whether some alteration in the statistical design is necessary.

Many statistical tests embody generic assumptions regarding the distribution from which the original data are drawn, and in application may embody further assumptions about the consistency, temporal continuity and regularity, and range of the specific test data. Ideally, the software used to perform the data reduction, tests for these assumptions and adjusts or suspends processing based on the fit of the test data to the design assumptions. Even more useful would be a real-time approach which would monitor and allow adjustment of test procedures to expand, contract, or modify the data collection process to meet the statistical requirements.

The global assumption implicit in many tests regards the distribution from which the test data are derived. Some tests perform reasonably well when the statistical distribution assumptions are not met, while others can yield seriously misleading results. At present, the fit of data to some distribution is performed using standard algorithms by a number of software packages. The potential for performance enhancement to real-time levels may exist through use of neural computer techniques to create a distribution recognition system which could function at data collection speeds. The initial project would be to train a neural computer network to recognize:

a. A static fit of a data set to a distribution.

b. A dynamic fit of an incrementally growing data set to a specified sample size and distribution requirement.

2.5.2 Subtest Selection. C³I system tests consist of 15 to over 100 subtests. Forty to 60 percent of the tests are system unique. The remainder are applications of a standard test criterion and methodology to a specific system. The unique tests are often identifiable in generic terms, but differ in detail to a degree which precludes standardization of the actual test procedure.

Selection of subtests for a system involves consideration of numerous factors and constraints. If the more obvious resource constraint problems are ignored, the remaining factors may be characterized as follows:

a. Category of test, i.e., developmental or operational, phase I, II or III;

b. Type of test, e.g., advanced development production qualification, engineering design, etc.

c. System application area, e.g., artillery, infantry, garrison or battlefield, air- or land-deployed, etc.

d. System characterization, e.g., electronic, mechanical, optical, aircraft ancillary, avionics, munitions, weapon system, radio, radar, active or passive, etc.

Some factors overlap, i.e., the factors are intermixed, and the system characterization may also include technologically unique components.

The selection of subtests is an activity calling on expertise from many areas in USAEPG, and involves considerable time and effort. The current problem is that any simplistic, rule-based approach must ignore many of the factors, while a more extensive approach would require extensive knowledge engineering for both development and maintenance. The resultant system would produce only a "straw man" selection for further refinement, in light of proponent and test agency resource constraints.

The proposed neural computer prototype in this domain would involve training a system to produce a selection of at least the standard subtests based on historical data regarding the system encoded in the United States Army Test and Evaluation Command and the Research, Development, Test, and Evaluation project numbers, the type and category of test from the test report cover sheet, and the actual tests selected from the test report table of contents.

The underlying hypothesis is that there is a gestalt, perhaps deeply buried, in the current mechanics of subtest selection, which a neural computing network could elicit from training examples of historical data on subtest selection.

2.5.3 Global Test Resource Estimation. As indicated in the subtest selection application, test selection is a complex process. Estimation of resources at a global level (i.e., total test time and number of test items) depends, at present, on the results of the subtest selection process.

The hypothesis is that there is a gestalt which might be elicited by examining the results of previous expert activity in the domain. Although the characterization data, available from the cover page and table of contents, will remain the same, the test report will need to be examined in greater detail to determine the total time and test item resources actually employed during the system test. The likelihood of useful gestalts emerging from training this prototype is somewhat greater than in the subtest selection case, since only the total test time and number of test items are desired, rather than a pattern of selection from among 50 to 100 standard subtests.

However, it may be more difficult to perform the analysis and extraction of historical resource data from the existing system test reports.

This prototype is designed to identify, based on preliminary system characterization, the test time and test item resources required for the global system classification. An additional function could identify potential impacts of deletion or addition of selected tests or subtests.

This page intentionally blank

SECTION 3. APPENDIXES

This page intentionally blank

APPENDIX A. REFERENCES

1. Executive Summary, DARPA Neural Network Study, 8 July 1988.

2. Methodology Investigation for Software Technology for Adaptable, Reliable Systems (STARS), Final Report: Test Methodology with Artifical Intelligence. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona, March 1987.

3. C. K. Ogden, Methodology Investigation Interim Report: Test Methodology with Artifical Intelligence. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona, October 1985.

4. C. Hall, Artificial Intelligence (AI) Test Officer Support Tool Project, Work Assignment Product (WAP). Trip Report, Comarco, 30 March 1988.

5. C. Klimasauskas, An Introduction to Neural Computing. NeuralWare Inc., 1988.

6. C. Klimasauskas, The 1987 Annotated Neuro-Computing Bibliography. NeuroConnection, 1987.

7. C. Klimasauskas, Teaching your Computer to Learn: Applications of Neural Computing. NeuralWare Inc., 1988.

8. D. E. Rumelhart, Parallel Distributed Processing Explorations in the Microstructure of Cognition, Volume I, Foundations. The MIT Press, 1987.

9. T. Kohonen, Self-Organization and Associative Memory. Springer-Verlage, 1984.

Recommended for Additional Reading:

10. J. L. McClelland, Parallel Distributed Processing Explorations in the Microstructure of Cognition, Volume II, Psychological and Biological Models. The MIT Press, 1987.

11. J. L. McCelland, Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises. The MIT Press, 1988.

12. J. A. Anderson, Neuro-Computing. The MIT Press, 1988.

This page intentionally blank

APPENDIX B. ACRONYMS AND ABBREVIATIONS

The following is a list of terminology used in this report, with the acronym/abbreviation associated with each term.

AI.....Artifical intelligence

BPN.....Backpropagation

CAD/CAM.....Computer-aided design/computer-aided manufacturing

C^3I.....Command, Control, Communications, and Intelligence

CPN.....Counterpropagation

ECR.....Embedded computer resources

KB.....Kilobyte

MB.....Megabyte

PC.....Personal computer

PDP.....Parallel Distributed Processing

PE.....Processing element

TECOM.....United States Army Test and Evaluation Command

USAEPG.....United States Army Electronic Proving Ground

This page intentionally blank

APPENDIX C. GLOSSARY

The following terms are identified and defined as they are used throughout this methodology report.

Activation Level

The minimum value required by the PE transfer function to produce an output from the PE.

Adaptability

The ability of neural computing systems to self-adjust. This ability is used for learning.

Adaptive Coefficient

The weighting value associated with each input to a PE. It weights the effect of that input on the PE's output. Adaptive coefficients can be self-adjusting; e.g., their values can be self-modified in response to external input. The process of self-adjusting is called learning.

Associative or Content-addressable Memory

Memory which allows retrieval of information by presentation of inexact or incomplete stored memory keys. It searches data on the basis of their contents rather than their location, and thus can be addressed using only a partial pattern or memory. When a neural network is stimulated with some fragment of a pattern in its associative memory, the network will respond with the entire memory or pattern.

Auto-association or Unsupervised Training

A means of training adaptive neural networks which requires unlabeled training data and no external teacher. Data is presented to the network and internal categories or clusters are formed which compress the amount of input data that must be processed at higher levels without losing important information.

Auto-associative Memory

Memory which is designed to transform an input pattern itself. If the input pattern is noisy, degraded, or incomplete, the memory will still recall the undegraded pattern.

Backpropagation (BPN)

A learning algorithm for updating weights in a multilayer, feed-forward neural network that minimizes the mean squared mapping error.

Bias

A constant input value applied to PEs to establish a reference level of operation.

Competition or Competitive Learning

A learning algorithm in which groups of PEs in a neural network compete to respond to an input pattern. The winner within each group is the one whose connections make it respond most strongly or activeiy to the pattern. The winner then adjusts its connections slightly toward the pattern that it has won.

Convergence

The process in which, after a finite number of presentations to the neural network input of a given set of training patterns, the values of the connection weights approach the set of values representing whatever computation or classification is contained in the input patterns.

Counterpropagation (CPN)

A type of neural network learning algorithm which does not require explicit tutoring of input-output correlations and spontaneously, through auto-association, self-organizes upon presentation of input information patterns. It is used in optimization and pattern classification problems.

Cross Talk

The overlap of input patterns in a neural network. This can result when a network does not have enough processing elements to allow one element to be reserved exclusively for every possible input pattern.

Distributed Memory

The independent memory of each PE in a neural network. This allows each processor to work on a small portion of the overall computational problem, thus distributing the load. Each entity or concept is represented by a pattern of activity distributed over many PEs, and each PE is involved in representing many different concepts.

Fan-out

The number of PEs directly excited by a given unit.

Feedback

A characteristic of a multilayer neural network with recursive connections that iterate over many cycles to produce an output. Contrast with "feed-forward".

Feed-forward

A characteristic of a multilayer neural network with connections exclusively from lower layers. In contrast to a feedback network, a feed-forward network operates only until its inputs propagate to its output layer. A multilayer perceptron architecture is a feed-forward neural network.

Full Connectivity

The condition in which each PE in a layer of a neural network is connected to each PE in another layer of the network.

Generalization

The ability of a neural computing system to generalize from the input/output training examples to produce a sensible output from a previously unseen input.

Gradient Descent Algorithm

A stochastic computational technique derived from statistical mechanics for finding near globally minimum-cost solutions to large optimization problems.

Hidden Units or Layers

Those PEs in multilayer network architectures which are neither the input layer nor the output layer, but are located between these and allow the network to undertake more complex problem solving, with nonlinear properties, than networks with no hidden units.

Input Pattern

A collection of input data items that are sent to the neural computer to act as the external stimulus to the network.

Interconnects or Connections

The unidirectional links or information channels between a neural network's PEs.

Knowledge Base

An unstructured set of facts and a set of inference rules for determining new facts.

Layer

A collection of PEs that use the same transfer and learning functions. Although the weights of individual PEs within a layer may vary, all have the same transfer function.

Learning

The process of adjusting the PE weights in response to external inputs.

Learning Rule, Function, or Algorithm

A first-order, ordinary differential, or difference equation governing the output state of a PE. The equation specifies how the adaptive coefficients or weights are self-modified in response to input signals and values supplied by the neural network's transfer function. This allows a PE's responses to input signals to change over time.

Least-mean-square Algorithm

A function which minimizes the mean squared error between the desired output of a neural network and the actual output.

Machine Learning

A research effort that seeks to create computer systems which can learn from experience.

Modality

A qualitative or quantitative weighting associated with a fact, element or knowledge, or conclusion. Examples from current expert systems include probability, certainty, utility, belief, desirability, and reliability.

Modal System

The set of techniques used by the inference engine to establish, propagate, and combine symbols or numbers representing modality values.

Network Architecture

The organization of a neural network, which encompasses the number of PEs, their neurodynamics, the configuration of their connections and layers, and the network or layer learning rules.

Neurodynamics

A mathematical representation of a PE and its connection which describes the operation of the PE.

Neural Computer

A hardware or software implementation of a neural network on a standard digital computer which allows software on the host computer to call neural network procedures. Neural computers allow neural networks to be integrated into almost any computer environment where their unique processing capabilities are needed.

Neural Network

A cognitive information processing structure based on models of brain functions. In an engineering context, a highly parallel dynamic system with the topology of a directed graph that can carry out information processing by its response to input data.

Parallel Processing

A computer system in which a program is executed concurrently on more than one processor, as opposed to serially on a single processor.

Pattern Recognition

A technique that classifies data into predetermined categories, using statistical methods, template comparisons, or learning algorithms.

Perceptron or Adaline (Adaptive Linear Neuron)

A member of a family of trainable pattern classifiers which distinguishes between patterns on the basis of linear discriminate functions.

Processing Element (PE)

The fundamental computational element in a neural network, which emulates the operation of neurons in living organisms. PEs are connected to each other via information channels called interconnects.

Resonant Network

A nonlinear neural network which permits a state of oscillation or feedback between the fields or layers in an associative memory.

Self-organization

The autonomous modification of the dynamics of a complete neural network, via learning in some or all of its PEs, to achieve a specified end capability or result.

Slice

A subgroup of a layer which contains a distinct logical collection of PEs.

Stochastic

A process involving a randomly determined sequence of observations. It implies randomness as opposed to a fixed rule or relation in passing from one observation to the next in order.

Training

The exposure of a neural network to a specified data set or information source environment, for the purpose of achieving a specified self-organization goal.

Training Set or Historical Training Data

The collection of data used in "training" a neural network to perform the desired pattern classification.

Transfer Function

The equation that defines how the output signal of a PE evolves in time as a function of its weighted input signals.

Vector

A quantity represented by an ordered set of numbers.

Weight

An adaptive coefficient that can be self-adjusted in response to each external input to a PE.